Automatic Image Colorization using CNNs

Alexandre Dalban & Wacil Lakbir

Automatic Image Colorization using CNNs

In this project, we tackle automatic image colorization: given a grayscale image, the goal is to predict a plausible color version using convolutional neural networks.

Task:
- Input: grayscale image (1xNxN)
- Output: color image (3xNxN)

Dataset:
Pexels API: Hand-Selected
- 512x512 images
- 96x96 (rescaled 512) images
- 5800 train / 1000 test
- High colour images
- People, Places, Things

Samples from the Dataset









The Naive Approach and the Averaging Trap

Problem Statement:

Given a grayscale image pixel with intensity value $I_{gray} \in [0, 255]$, a naive regression approach would learn a function $f: I_{gray} \rightarrow RGB$ by minimizing pixel-wise L2 loss (MSE) on training data. When multiple ground truth colors map to similar grayscale values, the network optimizes for the mean color across all samples.

- Sample 1: $I_{gray} = 128 \rightarrow RGB_1 = (255, 0, 0)$ [red]
- Sample 2: $I_{gray} = 128 \rightarrow RGB_2 = (0, 255, 0)$ [green]
- Sample 3: $I_{gray} = 128 \rightarrow RGB_3 = (0, 0, 255)$ [blue]

MSE loss: $L = |f(128) - RGB_{true}|^2$

- To minimize expected loss: $f(128) = E[RGB \mid I_{gray} = 128] = (85, 85, 85)$ [brown]

The network learns conditional mean prediction: for ambiguous grayscale values, MSE regression converges to the average of all possible colors.

The network cannot distinguish between distinct color modes when they share grayscale intensity.

Our Approach: L1 loss & Model Architecture

No loss function alone can fully solve the color ambiguity problem when multiple plausible colors share the same grayscale intensity.

1. Combined Loss Function:

- $L\_total = p \times L_1(\hat{Y}, Y) + (1-p) \times MSE(\hat{Y}, Y)$
- L1: Encourages sharper predictions (minimizes to conditional median)
- MSE: Provides stability and spatial smoothness

L1 operates independently per RGB channel. For example, with red (255,0,0), green (0,255,0), and blue (0,0,255) mapping to the same gray value, L1 predicts the median per channel: (0,0,0) [black], while MSE predicts the mean: (85,85,85) [brown]—neither picks an actual pure color.

2. U-Net Architecture:

- Encoder extracts semantic context:
- Skip connections preserve spatial details

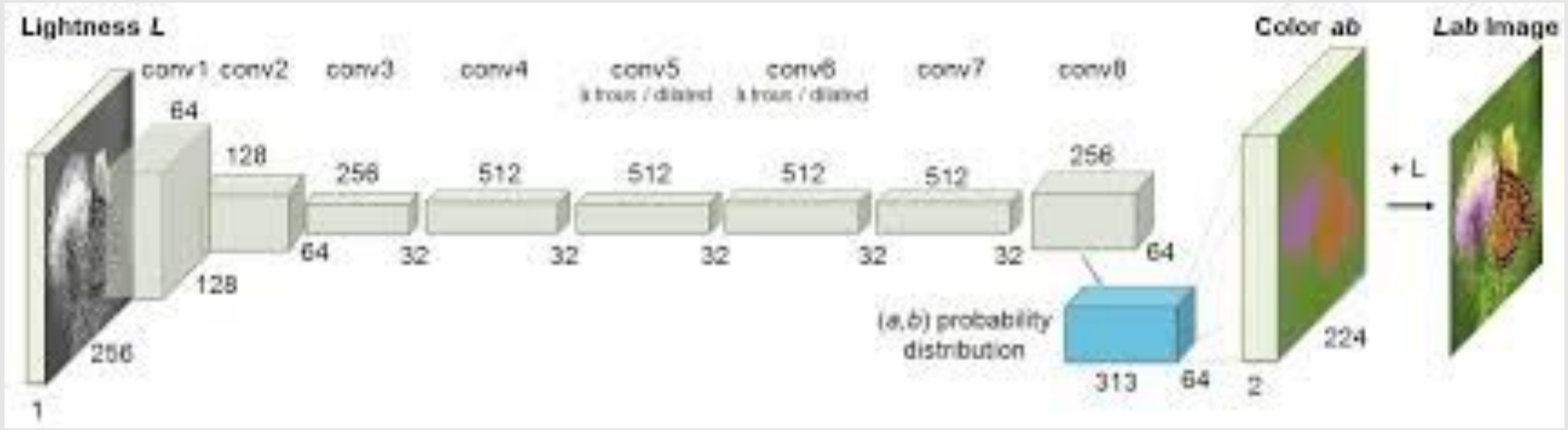
Baseline Idea: Simple CNN Encoder–Decoder

Task: Grayscale 1×96×96 → RGB 3×96×96 colorization

Encoder: 4× [Conv + BN + ReLU + MaxPool] → downsampling 96→48→24→12→6, channels 1→64→128→256→512

Decoder: 4× [ConvTranspose + BN + ReLU] → upsampling 6→12→24→48→96, channels 512→256→128→64→32, final Conv(32→3) + Sigmoid

~4.3M trainable parameters

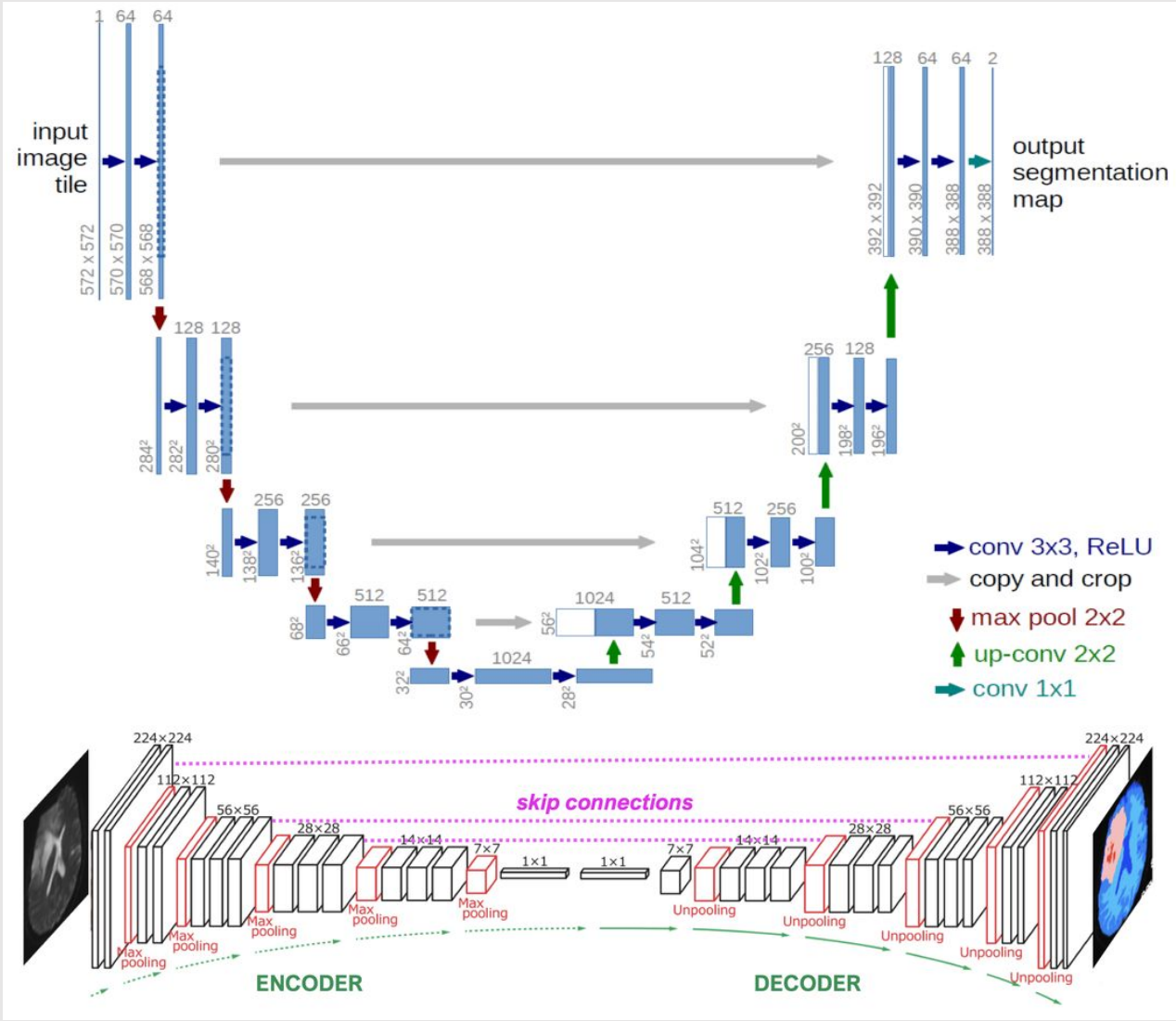


Limitations

- Strong compression → loss of fine spatial details
- Colors can be **blurry**, **desaturated**, or **misaligned** with objects

Improved Architecture: U-Net

- Encoder-Decoder Structure:
  - Downsampling path captures semantic features
  - Upsampling path reconstructs spatial resolution
- Skip Connections: Direct pathways from encoder to decoder at each scale preserve fine details and color boundaries
- Multi-Scale Processing: Captures both global context (what objects are present) and local texture (fine color details) simultaneously
- Prevents Color Bleeding: Skip connections + multi-scale features ensure colors don't spill across object boundaries



Another Approach to the Input: LAB

RGB approach: Input grayscale (1 ch) → Output RGB (3 ch) — must predict brightness + color

LAB approach: Input L channel (1 ch) → Output AB channels (2 ch) — only predict color!

- L = Lightness (0-100) — the grayscale image itself
- A = Green ↔ Red axis (-128 to +127)
- B = Blue ↔ Yellow axis (-128 to +127)

Data Pipeline:

1. Load RGB image from ImageNet (512×512×3)
2. Convert RGB → LAB color space
3. Extract L channel → normalize to [0,1] → model input
4. Extract AB channels → normalize to [-1,1] → training target
5. Model predicts AB (512×512×2)

Reconstruct: [L, predicted_AB] → convert LAB → RGB for display

Training Setup: U-Net LAB

Model Specifications:

- Total parameters: ~20.5 million
- Architecture depth: 4-level U-Net encoder-decoder
- Input resolution: 512×512×1 (L channel)
- Output resolution: 512×512×2 (AB channels)

Training Hyperparameters:

- Optimizer: Adam with learning rate = 1e-4
- Batch size: 16
- Epochs: 15

Loss Function: L_total = 0.5 × MSE(pred_AB, true_AB) + 0.5 × L1(pred_AB, true_AB)

Training Insights:

- Larger batch size (16 vs 8) → more stable gradients, faster convergence
- 15 epochs → allows model to learn complex color-semantic relationships
- Combined loss → balances smoothness with color saturation

Training Setup: U-Net RGB

Model Specifications:

- Total parameters: ~4.5 million
- Architecture depth: 4-level U-Net encoder-decoder
- Input resolution: 96×96×1 (Grayscale RBG)
- Output resolution: 96×96×3 (RGB)

Training Hyperparameters:

- Optimizer: Adam with learning rate = 1e-4
- Batch size: 512
- Epochs: 20

Loss Function: L_total = 0.7 × MSE(pred_AB, true_AB) + 0.3 × L1(pred_AB, true_AB)

Training Insights:

- Larger batch size (512) as input dim is smaller
- Higher MSE weight →used to test the difference

PSNR (Peak Signal-to-Noise Ratio)

Measures image reconstruction quality by comparing the predicted image to ground truth.

PSNR = $10 \times \log_{10}(MAX^2 / MSE)$

- MAX = maximum pixel value (1.0 for normalized images, 255 for 8-bit)
- MSE= mean squared error between predicted and ground truth

For us: PSNR = 10 * log10(1.0 / mse) (since images are normalized to [0,1])

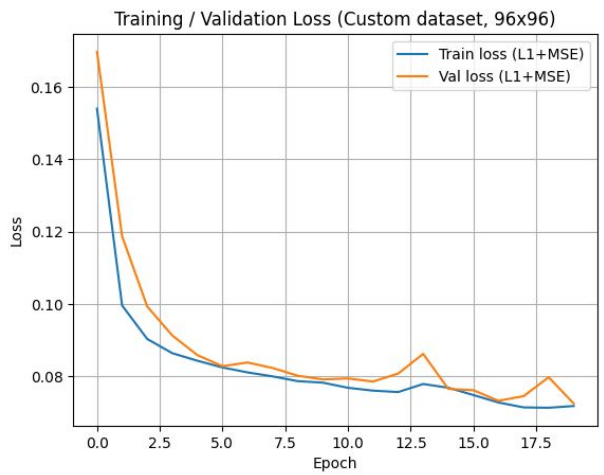
- Typical range: 20-40 dB for image reconstruction tasks, Higher PSNR = better
- Below: 20 dB = poor quality, Above: 30+ dB = good quality

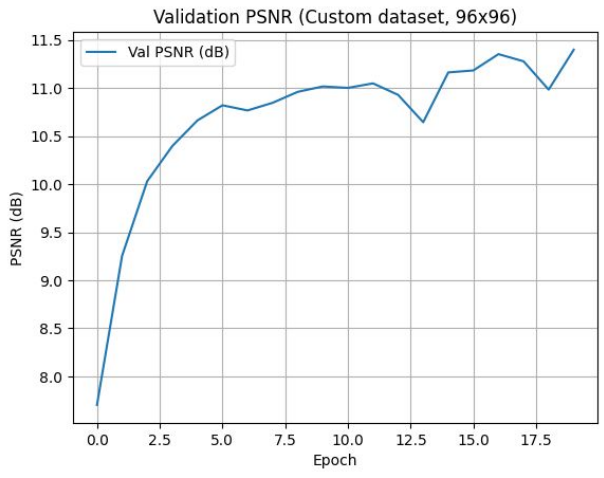
It's a pixel-level metric so it correlates with MSE but expressed in decibels

- Advantages: Easy to compute, interpretable, and commonly used in image processing benchmarks.
- Limitation: PSNR doesn't always correlate perfectly with perceptual quality—two images with same PSNR might look different to humans (especially for colorization, where color accuracy matters more than pixel-perfect MSE).

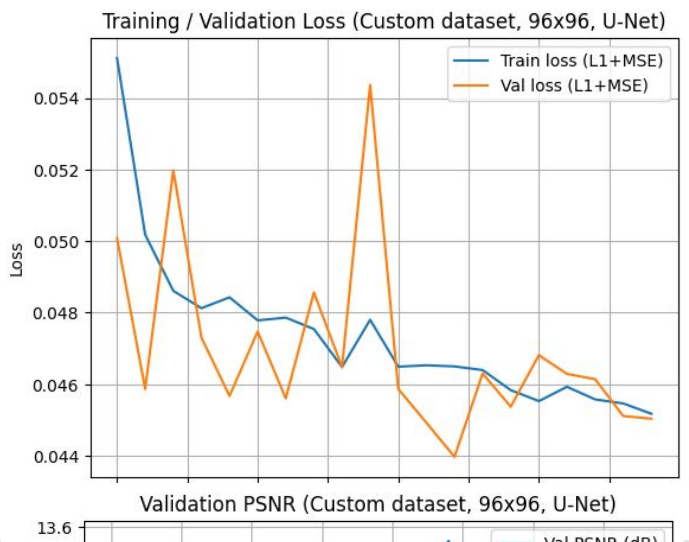
Quantitative results (MSE & PSNR)

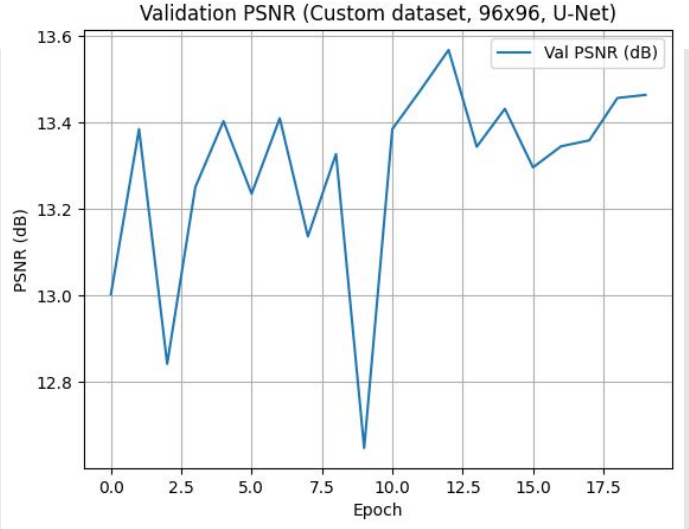
CNN 96: RGB



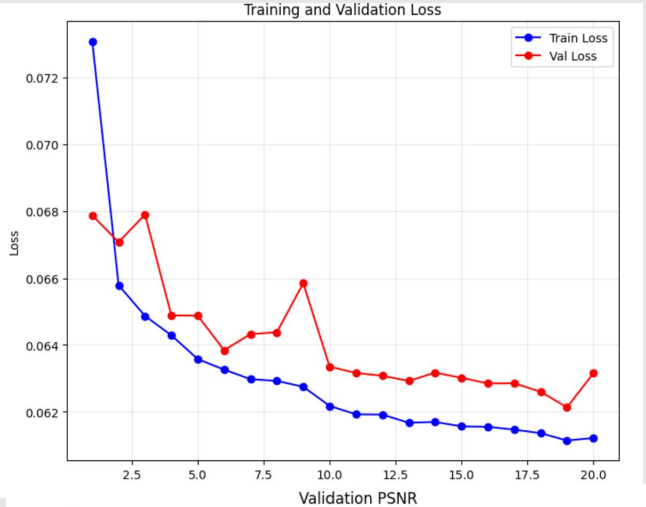


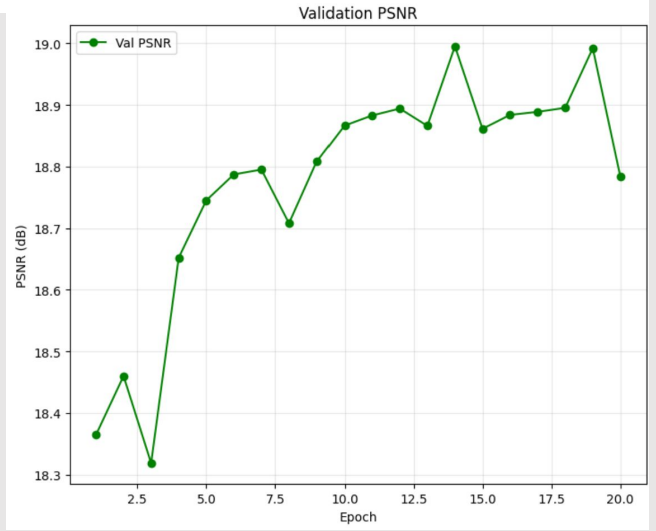
U-NET: RGB





U-NET: LAB

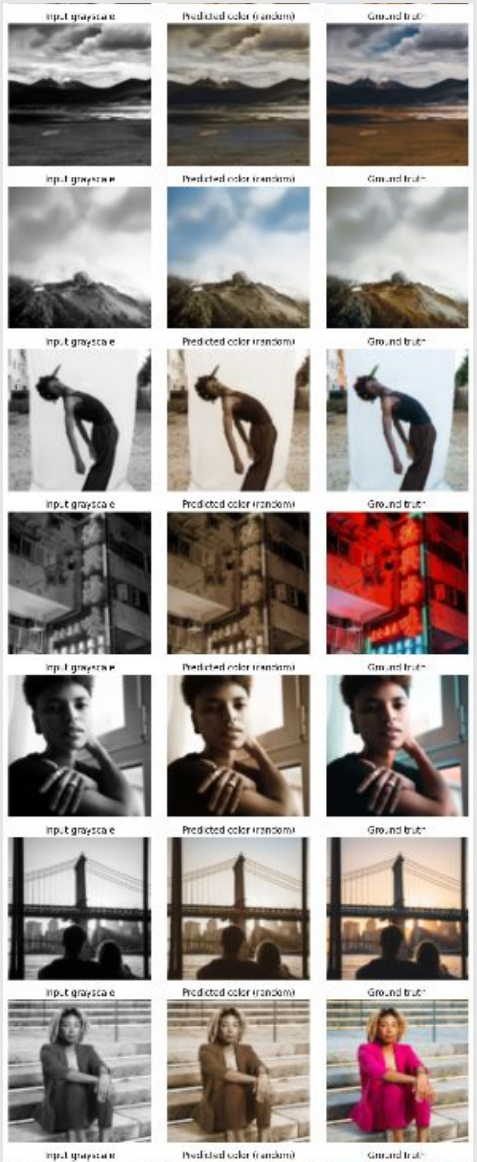




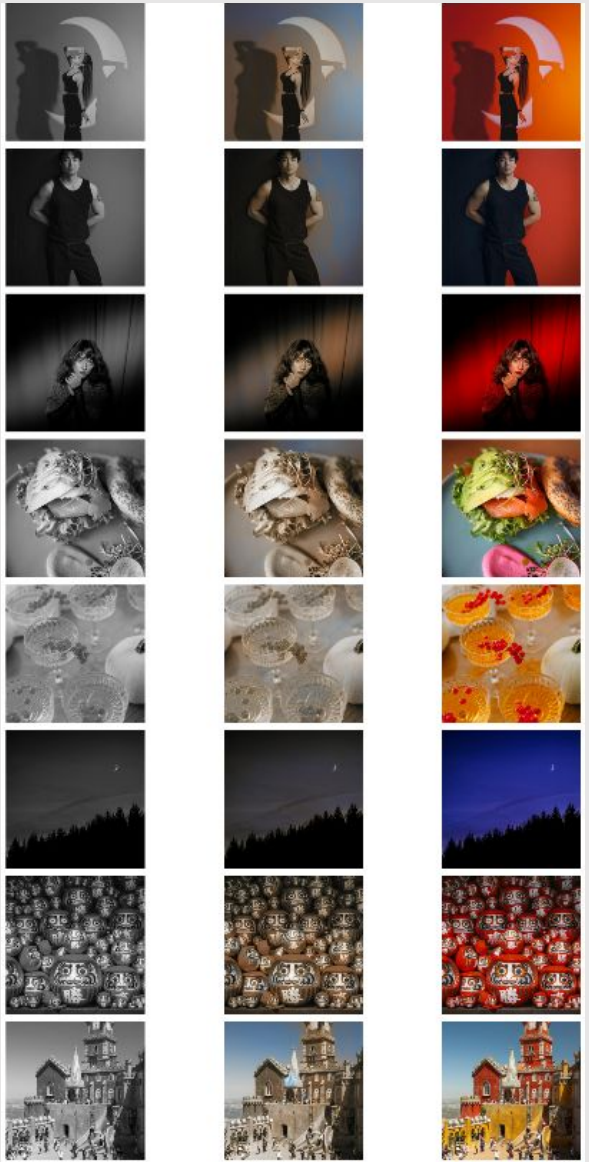
Visualizing results

CNN: RGB

U-NET: RGB



U-NET: LAB



Limitations

Despite using L1+MSE losses, U-Net features, and LAB space, the classic “brownish/low-saturation” problem still persists. These techniques can reduce averaging but cannot fully eliminate color ambiguity when many plausible colors map to the same grayscale region.

Adding extra terms like colorfulness loss often backfires, the model simply spray saturated colors to minimize the loss instead of producing meaningful, coherent colors.

Modern colorization systems avoid this by using color queries: instead of directly regressing pixel colors, the network predicts discrete color distributions, then selects or samples colors based on semantic understanding. This lets the model represent multiple possible color modes without averaging them together, avoiding the brown-collapse problem entirely.